(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 643 653 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
05.04.2006 Bulletin 2006/14

(51) Int Cl.:
H03M 13/00 *(2006.01)*

(21) Application number: **04255958.3**

(22) Date of filing: **29.09.2004**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PL PT RO SE SI SK TR
Designated Extension States:
AL HR LT LV MK

(71) Applicant: **LUCENT TECHNOLOGIES INC.**
Murray Hill, New Jersey 07974-0636 (US)

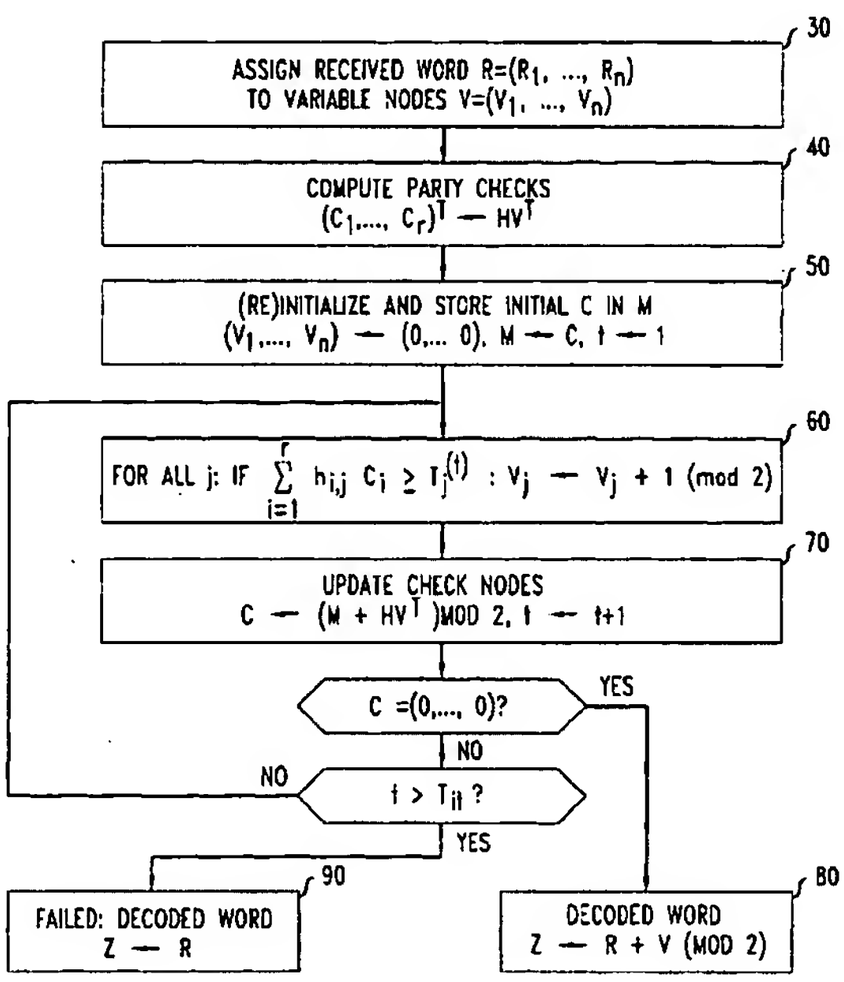
(72) Inventors:
- **Benet, Raul**
  08330 Premia de Mar (ES)
- **de Lind van Wijngaarden, Adriaan**
  Union
  NJ 07974 (US)
- **Dohmen, Ralf**
  78050 VS-Villingen (DE)
- **Richardson, Thomas**
  Essex
  NJ 07079 (US)
- **Urbanke, Rudiger**
  Lausanne (CH)

(74) Representative: **Sarup, David Alexander et al**
Lucent Technologies NS UK Limited
5 Mornington Road
Woodford Green, Essex IG8 0TU (GB)

(54) Iterative decoding of low-density parity-check (LDPC) codes

(57) An LDPC decoder has been designed that has reduced power requirements. After an initial parity check, the decoder holds in a register a flip indicator array with 0-values for those bits of the received codeword that satisfy a parity check, and 1-values for those bits that fail the parity check. In subsequent iterations, updates are performed on these bits rather than directly on the data bits. As a consequence, power requirements will be reduced.

FIG. 2



EP 1 643 653 A1

Printed by Jouve, 75001 PARIS (FR)

Description

Field of the Invention

[0001] This invention relates to forward error correcting codes and to decoders that effectively retrieve and restore the encoded information through the use of such codes.

Art Background

[0002] In modem technologies related to high-speed digital communication, as well as those related to the reproduction of recorded audio, video, or other data, it is often desirable to restore information that has been corrupted due to signal degradation in transmission, or due to damage suffered by the recording medium. Various methods have been used to encode signals with redundant information included in the encoded bit stream. At the receiver, the redundant information can be used in appropriate algorithms for correcting errors in the received signal. Such methods, which do not require any feedback from the receiver to the transmitter, are referred to as channel coding or "forward error control" (FEC) coding.

[0003] In high-speed optical communications, for example, it has proved useful to employ algebraic codes such as BCH codes and Reed-Solomon codes for forward error correction. These codes correct, for a given length n and dimension *k*, a predefined number of errors, *e*. The encoding and decoding complexity increases significantly with increasing block length *n* and redundancy *(n-k)*.

[0004] Although such codes are useful, there has been a need for alternative FEC codes, which are highly effective in correcting errors and have a low encoding and decoding complexity.

[0005] One useful alternative to algebraic codes is provided by so-called low-density parity-check (LDPC) codes. LDPC codes are defined by a set of binary parity-check equations. An LDPC code of length *n* and *r* parity-check equations can be specified as the set of all binary vectors x of length *n* that fulfill a linear algebraic equation $\mathbf{H}\,x = 0$, where **H** is a $r \times n$ binary matrix with entries $h_{i,j}$ and 0 denotes the null-vector of length *r*. This matrix has the property that it has a low density of ones. Each row corresponds to a grouping of selected bits of the data word, together with selected check bits, and it has the property that the sum of the values of the bits associated by each grouping is 0 modulo 2.

[0006] Systematic LDPC codes are a type of FEC codes in which the codeword is formed by appending check bits to the "data word," i.e., to the sequence of bits representing the original data prior to coding, and the values of the check bits are adjusted so that the sum of the bits associated by each grouping is 0 modulo 2. Systematic codes have the property that the bits of the data word are not modified and can be extracted directly from the codeword.

[0007] In the decoders for LDPC codes that have hitherto been known, data and check bits of the received codeword are held in a register. In each of potentially many iterations, the values of these bits are updated in accordance with the results of a parity check. After some number of iterations, this process tends to converge to a state in which the positions in the register that correspond to uncorrupted bits are storing the as-received values of those bits, whereas those positions that correspond to corrupted bits are storing the complements of the as-received values.

[0008] Although such decoders are useful, they could be made even more attractive by reducing their power requirements. The amount of power that is needed for a calculation in an Application Specific Integrated Circuit (ASIC) is typically directly related to the switching activity inside the circuit. Complementary Metal Oxide Semiconductor (CMOS) technologies and Bipolar CMOS (BiCMOS) technologies, commonly used for ASICs, draw a significant power supply current during logic transitions, and a negligibly small leakage current otherwise. This is an attractive low-power feature that can be exploited by minimizing the switching activity inside the circuit.

Summary of the Invention

[0009] We have developed an LDPC decoder that has reduced power requirements. After an initial parity check, our decoder holds in a register 0-values for those bits of the received codeword that satisfy a parity check, and 1-values for those bits that fail the parity check. In subsequent iterations, updates are performed on the bits in this parity register rather than directly on the data bits. Because most bits will, in fact, be received without corruption, the register will be populated mostly with 0-values and the switching activity will be low. As a consequence, power requirements will be reduced.

Brief Description of the Drawing

[0010]

FIG. 1 is a graphical representation of an example LDPC code.
FIG. 2 is a flowchart of a procedure according to the present invention in an exemplary embodiment.

Detailed Description

[0011] FIG. 1 illustrates a (3,4) regular LDPC code of length 20 that is defined by the matrix

$$\mathbf{H} = \begin{bmatrix}
1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1\\
0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0\\
0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0\\
0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
1&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&1\\
0&1&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&0\\
0&0&1&0&0&1&0&0&0&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&1&0&0&1&0&0&0&0&0&0&0&1&0&0&1&0&0\\
0&0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0&1&0\\
1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0\\
0&1&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&0\\
0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&1\\
0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0
\end{bmatrix}$$

[0012] That is, every codeword of this LDPC code is twenty bits long, and each bit of the codeword must satisfy three parity conditions, which in this example are linear constraints. Each of the linear constraints involves four bits of the codeword.

[0013] FIG. 1 represents the code in terms of a bipartite graph consisting of nodes interconnected by edges. Nodes $V = V_1 \ldots V_{20}$, which belong to array 10, are referred to as "variable nodes." These values may be stored in, e.g., a register. Initially, each node is associated with the binary value of a corresponding bit of the received word $R = R_1 \ldots R_n$. Nodes $C = C_1 \ldots C_{15}$, which belong to array 20, are referred to as "check nodes." Associated with each check node is the sum, modulo 2, of all variable nodes that are its neighbors; that is, of all variable nodes that are connected to it by an edge, i.e., $C = \mathbf{H}V^T$. The binary values associated with the check nodes may be stored, e.g., in a register. It will be seen that because the code is (3, 4) regular, every variable node has degree 3 and every check node has degree 4.

[0014] To illustrate the decoding procedure, consider the situation where the received word R is 00000111111111100000. The first step is to assign R to V. Next, the check nodes C are computed. It will be seen that all of the parity sums in array 20 have the value 0. Thus, the codeword represented by array 10 satisfies the parity condition. Hence, in this case $C = \mathbf{H}V^T = 0$. The decoding process is complete and the decoded word $Z = V$ is a valid codeword.

[0015] The operation of the decoder is now explained for the situation where the received word R is 00000111111011100000. This example may represent the situation in which a codeword has been received with an error at bit 12. That is, when compared to the previous example, the bit at node V_{12} has been changed from 1 to 0. As a consequence, the parity sums at nodes C_5, C_6, and C_{14} have changed from 0 to 1.

[0016] In accordance with the operation of the decoder, the parity check bits are passed from each check node back to its neighboring variable nodes. At each variable node i, the incoming parity sums are added and compared with a threshold value $T_i^{(t)}$, where t indicates the iteration number. Let $S = (S_1,\ldots,S_n)$ denote the parity sums for each variable node, such that S_i, where $1 \le i \le n$,. represents the result, at each respective variable node, of adding up the parity sums that have been passed back. For this example, S is (1,0,0,1,1,1,0,1,1,0,0,3,0,0,0,1,0,1,0,1). The respective values of C, S and V are specified for each iteration in Table 1.

Table 1

t	$T(t)$	$C= C_1...C_r$	$S=(S_1,...,S_r)$	$V = V_1...V_n$
0	-			00000111111011100000
1	3	000011000000010	(1,0,0,1,1,1,0,1,1,0,0,3,0,0,0,1,0,1,0,1)	00000111111111100000
2	3	000000000000000	(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)	00000111111111100000

[0017] It follows that the total stored at node S_{12} is 3, whereas the totals stored at all other variable nodes are less. A threshold test is applied to the total stored at each node, using, e.g., a comparator circuit adjusted to an appropriate threshold. If the total stored at any variable node equals or exceeds the threshold, the bit stored at the corresponding position in array 10 is replaced by its binary complement. Thus, if the threshold in the present example is set at 3 for all nodes, the bits of the codeword at variable nodes 1-11 and 13-20 will be unchanged, but the bit at node 12 will be reversed. The next step is to determine the parity check sums for the updated values at the variable nodes. It can be verified that in this example the parity sums $C_1...C_r$ are equal to the all-zero vector and therefore the variable nodes $V_1...V_n$ now contain a valid codeword.

[0018] In general, the process described above will be iterated. Different thresholds may be used for different variable nodes and for different iterations. It is generally beneficial for the threshold to slightly vary with increasing iteration number. The thresholds may be pre-set or adaptive. After each iteration, the variable node bits, as represented in array 10, are updated. The procedure continues until the parity-sums $C_1...C_r$ become equal to the all-zero vector, after which the sequence stored in the variable nodes $V_1...V_n$, a valid codeword, are output. If the parity-sums are not satisfied after a certain maximum number of iterations, the decoder stops and typically outputs the received word to avoid introducing extra errors.

[0019] We have found that a modification of the above procedure is useful for reducing power consumption in the digital processing device in which information in the array of variable nodes 10 and the array of check nodes 20 is stored.

[0020] According to our modified procedure, array 10 is, as before, an actively updated array of variable nodes, which initially stores the received word. However, the functionality of these nodes changes during the first iteration. After the first and every subsequent iteration, the array 10 instead stores a flip indicator for each respective bit of the codeword. Accordingly, array 10 will in this context be referred to as the "flip-indicator" array, which can actually be stored at the same memory locations as the variable node array V.

[0021] The modified procedure will now be described in further detail with reference to FIG. 2. At block 30 of FIG. 2, the array of variable nodes 10 (as seen in FIG. 1) are loaded with the bits of the received word R, as described above. At block 40, parity sums $C = \mathbf{H}V^T$ are computed at the check nodes, as described above.

[0022] At this point, the procedure takes on certain modified features. Array 10, which will hereafter function as a flip-indicator array, is initialized to contain all zero entries, as indicated at block 50 of FIG. 2. In addition, the computed values of the check nodes $C_1 ...C_r$ are stored in an array $M = M_1..M_r$

[0023] Then, as indicated at block 60, the sums

$$S_j = \sum_{i=1}^{r} h_{i,j} C_i$$

are determined by ordinary and not by modular arithmetic. Each sum S_j is subjected to a threshold test as described above. If the sum S_j is greater or equal than the threshold $T_j^{(t)}$, the flip-indicator value V_j is inverted, i.e., V_j is updated according to $V_j \leftarrow V_j \oplus 1$, wherein the symbol $\oplus$ denotes the exclusive-or (XOR) operation, which is equivalent to $V_j \leftarrow V_j + 1$ modulo-two addition.

[0024] Then, as indicated at block 70, each check node $C_1...C_r$ is updated by adding to its initial value M_i, via an XOR operation, the new values $V_1 ...V_n$ sent to it by each variable node in the neighborhood of C_i. The resulting sum at each check node constitutes a new parity sum. If the parity checks $C_1...C_r$ are identical to the all-zero vector, the decoding operation is complete and the decoded word Z is, as indicated in block 80, equal to $R \oplus V$. Otherwise, control then returns to block 60, where the sums S_j are recomputed and the variable nodes $V_1...V_n$ are updated subject to the threshold test. The procedure continues until the parity checks $C_1...C_r$ become equal to the all-zero vector or the number of iterations t exceeds a predefined value T_{it}. If the latter is the case, and $C_1...C_r$ is not the all-zero vector, the result of the decoding process is inconclusive and the decoder has not been able to find a valid codeword. It then typically outputs, as indicated in block 90, the received word R to avoid the insertion of decoding errors.

[0025] As an example that illustrates the decoding process, we introduce the codeword Y=1010100111100111110.

Assume that during transmission, three errors occur at, respectively, positions 2, 5, and 15 of the codeword. Thus, the received word R will have the form R=11100001111001010110 . We decode using a threshold scheme ($T^{(1)}$, $T^{(2)}$, $T^{(3)}$, $T^{(4)}$, $T^{(5)}$) = (3, 2, 3, 3, 3) , where $T_j^{(t)} = T^{(t)}$ for every variable node j . The parity-check vector C takes the values 01101010110000 1. At the variable nodes, we obtain the sum vector S= (0,2,1,1,3,1,1,2,2,1,1,1,2,1,2,2,1,2,1,1). Applying the first threshold $T^{(1)} = 3$, we obtain the flip-indicator vector V= 00001000000000000000 . In subsequent iterations in which the respective thresholds are applied, we obtain the vectors at the variable nodes and the corresponding flip-indicator vectors listed in Table 2.

Table 2

t	$T^{(t)}$	$C = C_1 \ldots C_r$	$S = (S_1, \ldots, S_r)$	$V = V_1 \ldots V_n$
1	3	01101010110000 1	(0,2,1,1,3,1,1,2,2,1,1,1,2,1,2,2,1,2,1,1)	00001000000000000000
2	2	01100010100000 0	(0,2,1,1,0,0,1,1,1,1,0,0,1,1,2,2,0,1,0,1)	01001000000000110000
3	3	00100010000001 0	(0,1,1,1,0,0,0,1,1,1,0,1,1,0,1,3,0,0,0,0)	01001000000000100000

[0026] It can be easily verified that convergence is reached after three iterations, because at the end of the third iteration, the check node vector will be the all-zero vector. It is therefore not necessary to perform further iterations.

[0027] Note that one could as well perform the test for an all-zero vector just before the operation in block 60 instead of just after block 70 (as shown in FIG. 2).

[0028] It should be noted that the particular flow depicted in FIG. 1 serves only as an example to illustrate the new method.

[0029] For pedagogical purposes, the example given above has involved a relatively short code. In general, the longer the code, the better will be the performance of the decoder. In practice, code lengths may be as high as 30,000 or even more. However, in contrast to conventional approaches, in our approach the power consumption depends mainly on the error fraction and not directly on the data that is transmitted. As a consequence, very great efficiencies are achieved in high rate codes and in long codes when error rates are relatively low.

Claims

1. A decoding method, comprising:

   a) performing a parity check on the bits of a received word;
   b) responsively to the outcome of the parity check, computing flip decisions for the bits of the received word, and
   c) repeating (a) and (b) at least once, wherein in each repetition of (a) the parity check is responsive to the current set of flip decisions,
   CHARACTERIZED IN THAT:
   d) step (b) and each repetition thereof comprises setting a respective indicator bit for each bit of the received word, such that the setting of the indicator bit indicates whether the corresponding bit of the received word should be flipped; and
   e) each repetition of (a) takes as input the values of the indicator bits in lieu of bit values of the received word.

2. The method of claim 1, further comprising, after a final repetition of step (b), updating the received word according to the values of the indicator bits.

3. The method of claim 1, wherein each parity check comprises computing parity sums, each parity sum involves a plurality of bits of the received word, and each bit of the received word that is subject to a parity check is involved in a plurality of parity sums.

4. The method of claim 3, wherein:

   in the parity check corresponding to the first execution of step (a), each parity sum is obtained by summing bits of the received word and the resulting parity sums are stored; and
   in the parity check corresponding to each repetition of step (a), each parity sum is obtained by adding a sum of current indicator bits to one of said stored parity sums from said first execution of step (a).

5. The method of claim 1, wherein the first execution of (a) results in a plurality of computed check bits that are stored for later use, and in each repetition of (a), new parity bits are computed from the current indicator bits together with the stored computed check bits.

6. A decoder, comprising:

a) a codeword register for storing a received word;
b) means for computing parity sums, arranged such that each parity sum relates to a plurality of bits of the received word and each bit of the received word relates to a plurality of parity sums;
c) a summing circuit arranged to compute a total parity sum for each bit of the received word; and
d) a conditional circuit arranged to issue, for each bit of the received word, an indication that the state of said bit should be flipped if the pertinent total parity sum passes a threshold test;
CHARACTERISED BY:
e) an indicator register for storing a plurality of indicator bits, each said indicator bit corresponding to a respective bit of the received word, wherein the indicator register is arranged to be updated in response to the flip indications issued by the conditional circuit; and
f) a logic circuit arranged to flip selected bits of the stored received word, as indicated by the bit values in the indicator register.

7. The decoder of claim 6, further comprising a storage register for storing an initial plurality of parity sums, and wherein the means for computing parity sums are arranged to update the parity sums from the contents of the storage register together with current values of the indicator bits.

FIG. 1
(PRIOR ART)

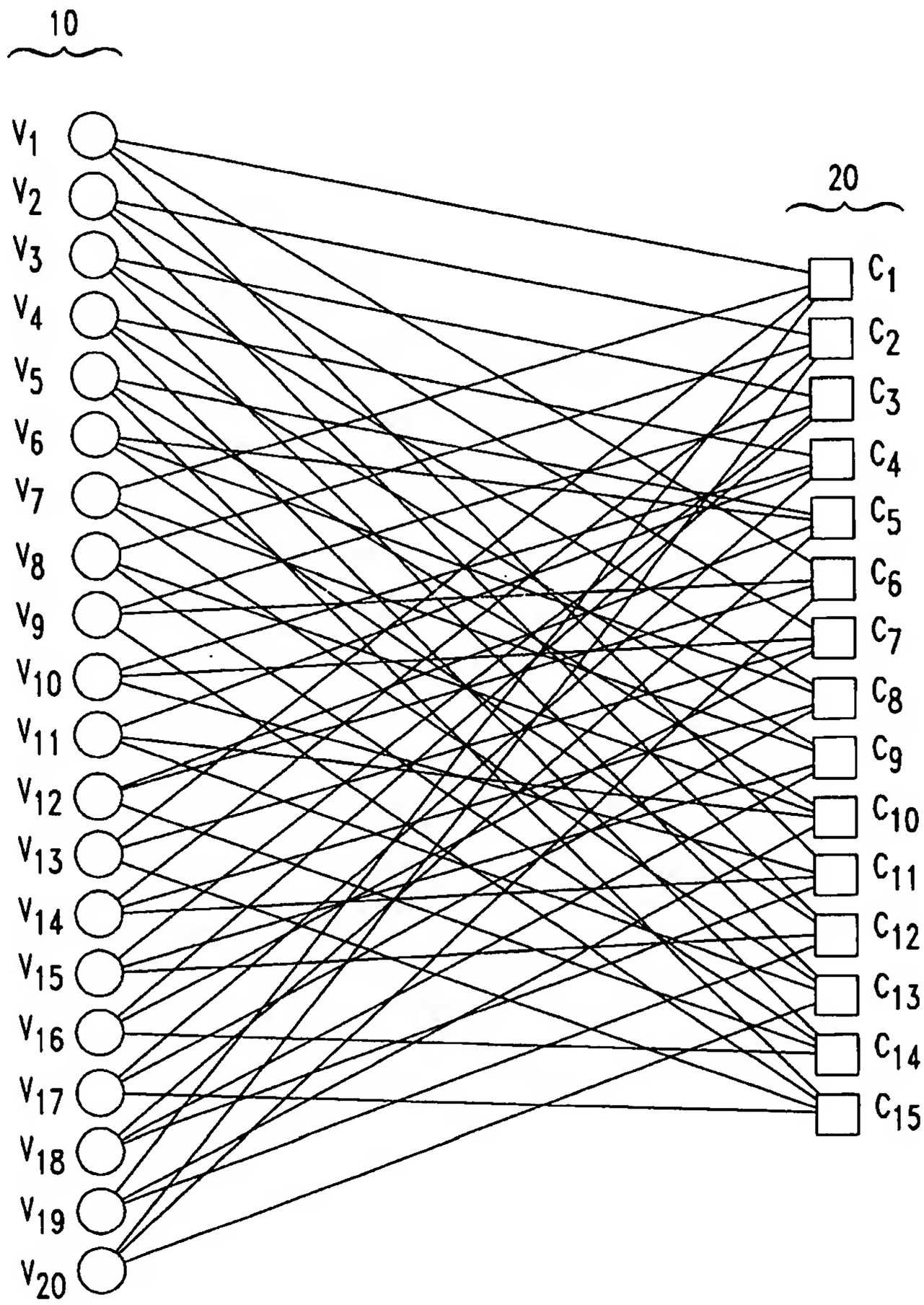
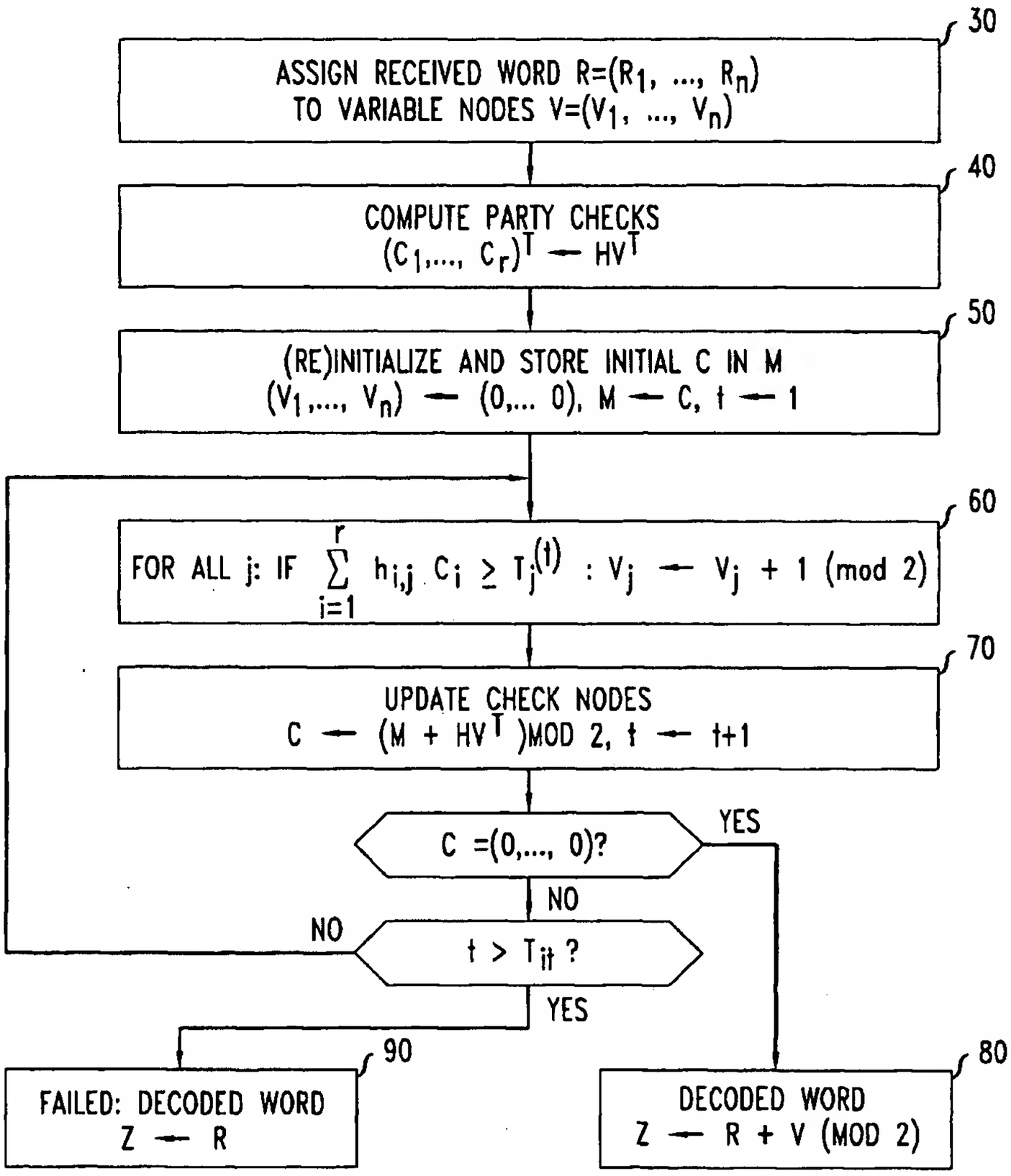


FIG. 2


30
ASSIGN RECEIVED WORD R=(R1, ..., Rn)
TO VARIABLE NODES V=(V1, ..., Vn)
40
COMPUTE PARTY CHECKS
(C1,..., Cr)T ← HVT
50
(RE)INITIALIZE AND STORE INITIAL C IN M
(V1,..., Vn) ← (0,... 0), M ← C, t ← 1
60
FOR ALL j: IF Σ hi,j Ci ≥ Tj(t) : Vj ← Vj + 1 (mod 2)
70
UPDATE CHECK NODES
C ← (M + HVT)MOD 2, t ← t+1
C =(0,..., 0)?
YES
NO
NO
t > Tit ?
YES
90
FAILED: DECODED WORD
Z ← R
80
DECODED WORD
Z ← R + V (MOD 2)

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 04 25 5958

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	KOU Y ET AL: "Low-density parity-check codes based on finite geometries: A rediscovery and new results" IEEE TRANSACTIONS ON INFORMATION THEORY, IEEE INC. NEW YORK, US, vol. 47, no. 7, November 2001 (2001-11), pages 2711-2736, XP002275913 ISSN: 0018-9448 * page 2718, left-hand column, paragraph 2 - right-hand column, paragraph 2 *	6	H03M13/00
A		1-5,7	
X	US 2004/148561 A1 (CAMERON KELLY BRIAN ET AL) 29 July 2004 (2004-07-29) * page 10, paragraph 127 - paragraph 138 *	6	
A		1-5,7	
A	GALLAGER R. G.: "Low-density parity-check codes" IRE TRANSACTIONS ON INFORMATION THEORY, IEEE INC. NEW YORK, US, January 1962 (1962-01), pages 21-28, XP000992693 * page 25, right-hand column, paragraph 3 *	1-7	
A	RICHARDSON T ET AL: "The renaissance of Gallager's low-density parity-check codes" IEEE COMMUNICATIONS MAGAZINE, IEEE SERVICE CENTER. PISCATAWAY, N.J, US, vol. 41, no. 8, August 2003 (2003-08), pages 126-131, XP001177711 ISSN: 0163-6804 * page 129, right-hand column - page 130, left-hand column *	1-7	
			TECHNICAL FIELDS SEARCHED (Int.Cl.7)
			H03M

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	23 November 2004	Offer, E

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 04 25 5958

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

23-11-2004

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004148561 A1	29-07-2004	NONE	

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82